Продолжаем знакомство с ООП языка C++ и вначале несколько подробнее поговорим о режимах доступа private и public. На предыдущем занятии мы с вами работали со следующим классом:

```cpp
class Point2D {
    int x, y;
public:
    void set_coords(int a, int b)
        {x = a; y = b;}
    void get_coords(int& a, int& b)
        {a = x; b = y;}
};
```

И я отмечал, что переменные x, y объявляются в приватной секции (private), так как класс (class), в отличие от структур, по умолчанию располагает все определения именно в приватной секции. Конечно, мы можем ее прописать и явно, например:

```cpp
class Point2D {
private:
    int x, y;
public:
    void set_coords(int a, int b)
        {x = a; y = b;}
    void get_coords(int& a, int& b)
        {a = x; b = y;}
};
```

Это будет одно и то же. Тем не менее, ключевое слово private записывать необходимо для обозначения начала приватной секции, например, после публичной:

```cpp
class Point2D {
public:
    void set_coords(int a, int b)
        {x = a; y = b;}
    void get_coords(int& a, int& b)
        {a = x; b = y;}
private:
    int x, y;
};
```

Если бы мы здесь не прописали private, то переменные x, y располагались бы в публичной (общедоступной) секции. Кроме того, в ряде случаев, явное указание private облегчает чтение текста программы.

Чем отличаются между собой секции private и public вы должны уже знать. Но я решил собрать всю эту информацию и еще раз повторить ее здесь. Начнем с публичной секции.

Все элементы (допустимые конструкции языка C++), объявленные в public, доступны как внутри объекта класса, так и за его пределами. Например, методы set_coords и get_coords совершенно свободно можно вызывать через объекты класса Point2D:

```language-cpp
Point2D pt;
pt.set_coords(1, 2);
```

И внутри (в телах) других методов класса:

```language-cpp
class Point2D {
private:
    int x, y;

public:
    void set_coords(int a, int b)
        {x = a; y = b;}

    bool set_coords_range(int a, int b, int min_coord = 0, int max_coord = 100)
        {
            if(a < min_coord || a > max_coord || b < min_coord || b > max_coord)
                return false;

            set_coords(a, b);
            return true;
        }

    void get_coords(int& a, int& b)
        {a = x; b = y;}
};
```

А вот **к элементам приватной секции (переменным x, y) можно получить доступ только изнутри класса и его объектов**. Например, через вызов методов set_coords и get_coords. В момент их вызова можно условно представить, что тело метода отрабатывает внутри класса (и то же самое, внутри объекта класса). Поэтому в теле метода мы имеем доступ ко всем

приватным определениям объектов класса. Но напрямую вне объекта при обращении к приватным полям компилятор выдаст ошибку:

```language-cpp
pt.x = 10; // ошибка
```

Вот в чем отличия приватной секции от публичной. И здесь важно понимать два момента:

1. Ограничение доступа private и public работает на уровне классов, а не на уровне объектов.
2. Ограничение доступа – это способ защиты программиста от возможного неправильного использования класса, а не от злоумышленников.

Давайте я поясню эти два утверждения. Первое. Что значит ограничение на уровне класса? Предположим, что мы бы хотели добавить метод в класс Point2D, который бы вычислял евклидово расстояние между текущей и второй переданной точкой. Этот публичный метод можно определить следующим образом:

```language-cpp
    double length_to(const Point2D& pt)
    {
            return sqrt((x-pt.x)*(x-pt.x) + (y-pt.y)*(y-pt.y));
    }
```

А, затем, вызвать:

```language-cpp
int main()
{
    Point2D pt, endp;

    pt.set_coords(1, 2);
    endp.set_coords(10, 20);

    double len = pt.length_to(endp);

    return 0;
}
```

Смотрите, в методе length_to мы имеем возможность работать с приватными переменными x, у двух разных объектов класса Point2D: текущего, через который был вызван метод; и переданного через параметр по ссылке. Это, как раз, говорит о том, что метод length_to, объявленный внутри класса,

получает доступ к приватным объявлениям любого объекта этого класса. **То есть, защита работает на уровне всего класса, а не отдельных объектов.**

Второе утверждение о защите программиста от возможных ошибок касается корректного (правильного) использования класса и его объектов в тексте программы. **Если что-либо помещено в приватную секцию, то разработчик класса предполагает обращения к этим полям через публичные методы, а не напрямую.** Причем, приватными могут быть не только переменные, но и некоторые методы класса. Всё, что находится в приватной зоне предназначено исключительно для внутреннего использования, но никак не для внешнего. Конечно, ушлый программист достаточно легко может обойти эту «защиту» и напрямую поменять значение любой приватной переменной. И программа даже может работать при такой реализации. Правда, здесь возможны следующие проблемы:

- программа на самом деле работает некорректно, просто программист этого пока еще не заметил;
- в будущем структура класса может поменяться и его прежнее использование с прямым доступом к приватным данным легко поломает внутреннюю логику работы этого класса.

Поэтому такие ушлые программисты либо быстро отучиваются делать такие «пакости», либо лишаются работы. Здесь следует придерживаться строгого правила: **все элементы приватных секций предназначены исключительно для внутренней логики работы класса и его объектов**. Вклиниваться в эту работу со стороны – строго запрещено. Это, как попытаться менять масло в работающем моторе. Ни к чему хорошему это не приведет. Также и с объектами классов – это отдельно и независимо работающие программные единицы и вмешиваться в их работу не нужно, допустимо только управление через публичные методы, которые для этого и предназначены. Ровно так, как руль или педали газа/тормоза предназначены для управления автомобилем. Ровно так следует воспринимать и публичные методы для управления состоянием того или иного объекта. Это принцип инкапсуляции, о котором мы говорили на вводном занятии. Через механизм защиты с помощью секций private и public она (инкапсуляция) и реализуется.